



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/687,233	10/15/2003	Martin Runte	6570P015	8107
45962	7590	12/08/2008	EXAMINER	
SAP/BSTZ			CHEN, QING	
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP			ART UNIT	
1279 OAKMEAD PARKWAY			PAPER NUMBER	
SUNNYVALE, CA 94085-4040			2191	
			MAIL DATE	DELIVERY MODE
			12/08/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/687,233

Applicant(s)

RUNTE ET AL.

Examiner

Qing Chen

Art Unit

2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 08 September 2008.
2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1,4-6,8,11-19,21,22,25,32,34 and 35 is/are pending in the application.
4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5) ☐ Claim(s) _____ is/are allowed.
6) ☒ Claim(s) 1,4-6,8,11-19,21,22,25,32,34 and 35 is/are rejected.
7) ☐ Claim(s) _____ is/are objected to.
8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date _____
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____

DETAILED ACTION

1. This Office action is in response to the amendment filed on September 8, 2008.
2. **Claims 1, 4-6, 8, 11-19, 21, 22, 25, 32, 34, and 35** are pending.
3. **Claims 1, 6, 21, 22, and 32** have been amended.
4. **Claims 2, 3, 7, 9, 10, 20, 23, 24, 26-31, 33, and 36-38** have been canceled.
5. The objections to Claims 1, 6, 22, 24, 32, 34, and 35 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims. However, the objections to Claims 4, 5, 8, 11-19, 21, and 25 are maintained in view of Applicant's arguments and further explained hereinafter.

Response to Amendment

Claim Objections

6. **Claims 4, 5, 8, 11-19, 21, and 25** are objected to because of the following informalities:
 - **Claims 4, 5, 8, 11-19, 21, and 25** recite the category of invention "[t]he method." Applicant is advised to change this category of invention to read "[t]he computer-implemented method" for the purpose of providing it with proper explicit antecedent basis and/or keeping the claim language consistent throughout the claims.
 - **Claim 25** contains a typographical error: Claim 25 should depend on Claim 22, not Claim 24.Appropriate correction is required.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. **Claims 1, 4, 32, and 34** are rejected under 35 U.S.C. 102(b) as being anticipated by US **2002/0072928** (hereinafter “**Sundararajan**”).

As per **Claim 1**, Sundararajan discloses:

- automatically detecting a change introduced into a software object of a first software subsystem, where the first software subsystem stores the software object from which instances of the software object are created, wherein the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem, where the first software subsystem is located at a server and the second software subsystem is located at a client (see Figure 1; Paragraph [0016], “An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer.”; Paragraph [0020], “Changes introduced to components existing in the production environment can include new software installations, software upgrades, hardware installations and hardware upgrades.”; Paragraph [0029], “In an example of a component change management system according to the present invention, an e-commerce

application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features (the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem).”; Paragraph [0045], “At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b and/or 14c (where the first software subsystem stores the software object from which instances of the software object are created). At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component (automatically detecting a change introduced into a software object of a first software subsystem).”);

- accessing a compatibility changes database in response to detecting the change, where the compatible changes database indicates changes defined to be compatible with the software objects of the second software subsystem (*see Paragraph [0038], “The component change manager 12 further includes a database 25 having a plurality of records A, B and C defined therein. More specifically, the records A, B and C each include a plurality of data fields and associated control fields related to inter-relationships of components located on each of the e-commerce servers 14a, 14b and 14c.”; Paragraph [0046], “After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received*

the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships.”);

- determining, based on accessing the compatible changes database, whether the change is compatible with the software objects of the second software subsystem, including determining whether the change is predefined as compatible (*see Paragraph [0046], “If the component revision is verified as compatible, at step 180, the component revision is executed at step 190.”*); and

- implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem (*see Paragraph [0046], “If the component revision is verified as compatible, at step 180, the component revision is executed at step 190. This component revision becomes the current application configuration definition for the production environment hosted on the e-commerce server 14a, 14b and/or 14c.”*); otherwise,

- rejecting the introduced change and generating an error notification (*see Paragraph [0022], “Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration.”*; Paragraph [0046], “If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.”).

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Sundararajan further discloses:

- issuing a message that the change is not allowed if the change is not predefined as compatible (*see Paragraph [0022], "Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration."; Paragraph [0046], "If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt."*).

Claims 32 and 34 are article of manufacture claims corresponding to the computer-implemented method claims above (Claims 1 and 4) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1 and 4.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

10. **Claims 5 and 35** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **US 6,658,659 (hereinafter "Hiller")**.

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Sundararajan does not disclose:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller discloses:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible *(see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release. ")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller into the teaching of Sundararajan to include allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released *(see Hiller – Column 13: 6-12)*.

Claim 35 is rejected for the same reason set forth in the rejection of Claim 5.

11. **Claims 6, 13-15, 17-19, 22, and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **US 6,725,452 (hereinafter “Te’eni”)**.

As per **Claim 6**, Sundararajan discloses:

- identifying a subset of software objects of a first software subsystem that are stored in the first software subsystem and are to be instantiated in a second software subsystem to interact with software objects of the second software subsystem, where the first software subsystem is located at a server and the second software subsystem is located at a client (*see Figure 1; Paragraph [0016], “An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer.”; Paragraph [0020], “Changes introduced to components existing in the production environment can include new software installations, software upgrades, hardware installations and hardware upgrades.”; Paragraph [0029], “In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features (instantiated in a second software subsystem to interact with software objects of the second software subsystem).”; Paragraph [0045], “At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b*

and/or 14c (identifying a subset of software objects of a first software subsystem that are stored in the first software subsystem).);

- detecting a change introduced into a software object from the subset of software objects; and prior to allowing the change (see Paragraph [0045], “At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component.”),

- accessing a compatible changes database in response to detecting the change, where the compatible changes database indicates changes predefined to be compatible with the software objects of the second software subsystem (see Paragraph [0038], “The component change manager 12 further includes a database 25 having a plurality of records A, B and C defined therein. More specifically, the records A, B and C each include a plurality of data fields and associated control fields related to inter-relationships of components located on each of the e-commerce servers 14a, 14b and 14c.”; Paragraph [0046], “After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships.”);

- determining with a compatibility check, based on accessing the compatible changes database, whether the change is compatible with a second software subsystem, including determining whether the change is predefined as compatible (see Paragraph [0046], “If the

component revision is verified as compatible, at step 180, the component revision is executed at step 190.”); and

- issuing a notice indicating results of the compatibility check (see Paragraph [0022], “Configuration maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration.”; Paragraph [0046], “If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.”).

However, Sundararajan does not disclose:

- declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems.

Te’eni discloses:

- declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems (see Column 14: 32 and 33, “LOCKS a set of components that were locked by the user in order to present installation or removal thereof ...” and 46-53, “At step 90, LOCKS is examined to check whether LOCKS contains B.CI. If the result of step 90 is positive, at step 92 MODE is examined to check whether the value thereof is “interactive mode”. If the result is negative, RESULT is returned to the calling routine with the value of “failure”. If the result of step 92 is positive, the user’s permission is requested at step 96 to remove the lock from the component.” and 55-57, “In

contrast, if the user allows the removal of the lock from the component at step 100, a series of tests are made at steps 98, 104, and 106.”).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Te’eni into the teaching of Sundararajan to include declaring the subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined to be compatible or the changes are approved by an expert in compatibility between the software subsystems. The modification would be obvious because one of ordinary skill in the art would be motivated to resolve inter-component compatibilities among hardware and software entities (*see Te’eni – Column 15: 9-14*).

As per **Claim 13**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object is a function module (*see Paragraph [0016], “Representative components may include hardware, application modules and language interpreters.”*).

As per **Claim 14**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object is a data structure (*see Paragraph [0016], “Representative components may include hardware, application modules and language interpreters.”*).

As per **Claim 15**, the rejection of **Claim 13** is incorporated; and Sundararajan further discloses:

- wherein the software object includes an environment of the function module (*see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters."*).

As per **Claim 17**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object includes an interface and an environment of the interface (*see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters."*).

As per **Claim 18**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein a software object includes a program and an environment of the program (*see Paragraph [0016], "Representative components may include hardware, application modules and language interpreters."*).

As per **Claim 19**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein the detecting the change comprises automatically monitoring development of software code (*see Paragraph [0045], "At step 160, the operator can execute the revision of the*

current application configuration definition, which is associated with the business models or production environment, by installing a new component or revising an existing component.”).

As per **Claim 22**, Sundararajan discloses:

- monitoring a software development space to detect changes introduced into a subset of software objects stored in a first software subsystem that are to be instantiated in a second software subsystem, where the first software subsystem is located at a server and the second software subsystem is located at a client (see Figure 1; Paragraph [0016], “An exemplary relationship within an e-commerce application might include, but is not limited to, identified compatibility between a web application server and web server software, which both cooperate to present one or more web pages to an end user, such as a user of a user computer.”; Paragraph [0020], “Changes introduced to components existing in the production environment can include new software installations, software upgrades, hardware installations and hardware upgrades.”; Paragraph [0029], “In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features.”; Paragraph [0045], “At step 150, an operator of any one of the e-commerce servers 14a, 14b and/or 14c can elect to revise a current application configuration definition associated with the business models or production environments, which are hosted on any one of the e-commerce servers 14a, 14b and/or 14c. At step 160, the operator can execute the revision of the current application configuration definition, which is associated with the

business models or production environment, by installing a new component or revising an existing component.”);

- performing a global compatibility check of the subset of software objects of the first software subsystem by determining whether any changes were introduced into the subset of software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were unapproved changes introduced without obtaining prior approval, where performing the global compatibility check includes comparing a current version of software code of a software object in the software development space with a version of the software code of the software object at the time of a last global compatibility check (*see Paragraph [0022], “A representative corrective action may include restoring a specified version of a component to a previous condition or status upon system detection of an error in a pre-determined relationship (wherein the introduced changes were unapproved changes introduced without obtaining prior approval).”; Paragraph [0023], “Verifying and maintaining the integrity of the application configuration definition may occur at random intervals or at predetermined intervals.” and “Revising existing components and/or adding new components, such as adding enhancements to specific components of a web application server, can expose a stable executing configuration to a potential operating error if the relationship of the components are not verified. Thus, verifying and maintaining the integrity of the application configuration definition is necessary before and after upgrading the e-commerce application with changes to the associated web application server software (performing a global compatibility check of the subset of software objects of the first software subsystem by determining whether any changes were introduced into the subset of software objects of the first*

software subsystem since the time of a last compatibility check). Subsequent verification and maintenance checkpoints may be required depending on the significance of the relationship to be validated and impact to the system if a relationship becomes corrupted between checkpoints.”;
Paragraph [0046], “After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships (comparing a current version of software code of a software object in the software development space with a version of the software code of the software object at the time of a last global compatibility check).”;

- identifying software objects of the second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software subsystem (*see Paragraph [0046], “After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships.” and “If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.”;* and

- issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change (*see Paragraph [0022], “Configuration*

maintenance may include, but is not limited to, notifying appropriate system operators of a relationship or inter-relationship inconsistency or taking corrective action to re-establish integrity of the configuration.”; Paragraph [0046], “If, at step 180, the component revision is not verified as compatible, the component revision is not executed at step 190, rather the operator is notified of the incompatible component revision attempt.”).

However, Sundararajan does not disclose:

- a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems.

Te’eni discloses:

- a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems (*see Column 14: 32 and 33, “LOCKS a set of components that were locked by the user in order to present installation or removal thereof ...” and 46-53, “At step 90, LOCKS is examined to check whether LOCKS contains B.CI. If the result of step 90 is positive, at step 92 MODE is examined to check whether the value thereof is “interactive mode”. If the result is negative, RESULT is returned to the calling routine with the value of “failure”. If the result of step 92 is positive, the user’s permission is requested at step 96 to remove the lock from the component.” and 55-57, “In contrast, if the user allows the removal of the lock from the component at step 100, a series of tests are made at steps 98, 104, and 106.”).*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Te'eni into the teaching of Sundararajan to include a subset of software objects frozen, where changes to the frozen software objects are not allowed unless the changes are predefined as compatible or the changes are approved by an expert in compatibility between the software subsystems. The modification would be obvious because one of ordinary skill in the art would be motivated to resolve inter-component compatibilities among hardware and software entities (see Te'eni – Column 15: 9-14).

As per **Claim 25**, the rejection of **Claim 22** is incorporated; and Sundararajan further discloses:

- wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem (see *Paragraph [0029]*, *"In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features."*).

12. **Claims 8, 11, and 12** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te'eni** as applied to Claims 6 above, and further in view of US 2004/0230952 (hereinafter "**Massaro**").

As per **Claim 8**, the rejection of **Claim 6** is incorporated; and Sundararajan further discloses:

- wherein frozen software objects are classified to include released objects and restricted objects based on a number of instances instantiated in the second software subsystem (see Paragraph [0042], “More specifically, the plurality of permutations of dynamic relationships or inter-relationships are determined by determining all possible combinations of components (released objects and restricted objects having a number of instances) that can be executed to carry out an e-commerce transaction as well as all possible combinations of component relationships and inter-relationships associated with each combination of components.”; Paragraph [0043], “For example, one permutation of an e-commerce transaction can include a user at user computer 20a, 20b and/or 20c communicating an e-commerce transaction to the e-commerce server 14a, 14b, and/or 14c ...”).

However, Sundararajan and Te’eni do not disclose:

- where released objects are defined as software objects having a number of instances instantiated in the second software subsystem exceeds a threshold number, and restricted objects are defined as software objects having a number of instances instantiated in the second software subsystem that does not exceed the threshold number.

Massaro discloses:

- using a threshold number to distinguish between versions of files both when the number of changes to a region is few and when the number of changes is large (see Paragraph [0010], “A method, apparatus, system, and signal-bearing medium are provided that in an embodiment compare versions and mark an entire region as changed when the number of changes to the region exceeds a threshold. In contrast, when the number of changes in a region does not exceed the threshold, the individual changes are marked. In this way, a user can

distinguish between versions both when the number of changes to a region is few and when the number of changes is large.”).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Massaro into the teaching of Sundararajan to include where released objects are defined as software objects having a number of instances instantiated in the second software subsystem exceeds a threshold number, and restricted objects are defined as software objects having a number of instances instantiated in the second software subsystem that does not exceed the threshold number. The modification would be obvious because one of ordinary skill in the art would be motivated to distinguish between components based on a usage limit.

As per **Claim 11**, the rejection of **Claim 8** is incorporated; and Sundararajan further discloses:

- wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object (*see Paragraph [0044], “The static and dynamic inter-relationships, including the plurality of permutations of dynamic inter-relationships, can be associated with data fields. The data fields are encoded control fields identifying the compatibility of one component (e.g., web server software) with another component (e.g., web application software). The data fields are stored in a record A, B, or C defined on the database 25, which database 25 is represented on the component change manager 12.”*).

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Sundararajan further discloses:

- wherein classification of each frozen software object is based on a number of instances of each frozen software object occurring in the second software subsystem (*see Paragraph [0042], "More specifically, the plurality of permutations of dynamic relationships or inter-relationships are determined by determining all possible combinations of components that can be executed to carry out an e-commerce transaction as well as all possible combinations of component relationships and inter-relationships associated with each combination of components."*; Paragraph [0043], *"For example, one permutation of an e-commerce transaction can include a user at user computer 20a, 20b and/or 20c communicating an e-commerce transaction to the e-commerce server 14a, 14b, and/or 14c ..."*).

13. **Claim 16** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te'eni** as applied to Claim 6 above, and further in view of **US 6,415,435 (hereinafter "McIntyre")**.

As per **Claim 16**, the rejection of **Claim 6** is incorporated; however, Sundararajan and Te'eni do not disclose:

- wherein a software object includes a class and an environment of the class.

McIntyre discloses:

- wherein a software object includes a class and an environment of the class (*see Column 1: 8-14, "The present invention relates generally to an improved data processing system*

and in particular to a method and apparatus for determining compatibility of different classes in a data processing system.”).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of McIntyre into the teaching of Sundararajan to include wherein a software object includes a class and an environment of the class. The modification would be obvious because one of ordinary skill in the art would be motivated to determine compatibility of different classes (see McIntyre – Column 1: 8-14).

14. **Claim 21** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Sundararajan** in view of **Te’eni** as applied to Claim 6 above, and further in view of **Hiller**.

As per **Claim 21**, the rejection of **Claim 6** is incorporated; however, Sundararajan and Te’eni do not disclose:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

Hiller discloses:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible (see Column 11: 21-25, “... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.”).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller into the teaching of Sundararajan to

include wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released (*see Hiller – Column 13: 6-12*).

Response to Arguments

15. Applicant's arguments filed on September 8, 2008 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a) As a first matter, if there was indeed something wrong with the preamble of the dependent claims, Applicants do not understand why it was not addressed in the previous Office Actions mailed prior to Applicants' filing for an RCE. Furthermore, Applicants are unaware of legal authority that defines how "a computer-implemented method" is a different statutory category from "a method," and request such legal authority be provided, or the objection be withdrawn. Furthermore, per MPEP 2173.05(e), "explicit antecedent basis" is not required where, as here, the claims are not unclear. Applicants have elected to use the preamble "The method..." in the dependent claims for simplicity and ease of readability. Applicants submit that no one of skill in the art would be confused into thinking that "the method" referred to in the dependent claims could be anything other than the "computer-implemented method" of the

independent claims. Therefore, the objection is without merit, and Applicants have chosen to leave the dependent claims unchanged.

Examiner's response:

a) Examiner disagrees. Applicant's arguments are not persuasive for at least the following reasons:

First, with respect to the Applicant's argument of why the objection was not addressed in the previous Office actions mailed prior to the Applicant's filing for an RCE, the Examiner respectfully submits that nowhere does the MPEP specify when a claim objection should be made during the prosecution. The claims are objected to when the Examiner deems the form of the claims to be improper and should not impact the Applicant's decision on filing for an RCE. Upon further consideration of the claims in response to the Applicant's filing of the RCE, the Examiner deemed the claims to be in improper form and thus, the claims were objected to for lacking proper explicit antecedent basis.

Second, with respect to the Applicant's request for such legal authority to be provided, as previously pointed out in the Non-Final Rejection (mailed on 06/06/2008) and further clarified hereinafter, the Examiner respectfully submits that providing the category of invention of the claims with proper explicit antecedent basis would improve the clarity or precision of the claim language used or, at the very least, keep the claim language consistent throughout the claims. Examiner further submits the relevant portions of MPEP § 2173.02 and 37 CFR 1.75 with emphasis added for purposes of convenience in discussion and illustration:

MPEP § 2173.02 Clarity and Precision

The examiner's focus during examination of claims for compliance with the requirement for definiteness of 35 U.S.C. 112, second paragraph, is whether the claim meets the threshold requirements of clarity and precision, not whether more suitable language or modes of expression are available. When the examiner is satisfied that patentable subject matter is disclosed, and it is apparent to the examiner that the claims are directed to such patentable subject matter, he or she should allow claims which define the patentable subject matter with a reasonable degree of particularity and distinctness. Some latitude in the manner of expression and the aptness of terms should be permitted even though the claim language is not as precise as the examiner might desire. Examiners are encouraged to suggest claim language to applicants to improve the clarity or precision of the language used, but should not reject claims or insist on their own preferences if other modes of expression selected by applicants satisfy the statutory requirement.

37 CFR 1.75 Claim(s).

(a) The specification must conclude with a claim particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention or discovery.

According to the section of the MPEP and the patent rule provided above, the Examiner would like to point out that a claim must particularly point out and distinctly claim the subject matter which the Applicant regards as the invention. In accordance with MPEP § 2173.02, the Examiner suggests providing the category of invention of the claims with proper explicit antecedent basis and/or keeping the claim language consistent throughout the claims to improve the clarity or precision of the claim language used. Hence, doing so would help the Examiner in reviewing the claims for compliance with 35 U.S.C. § 112, second paragraph.

Third, with respect to the Applicant's assertion that per MPEP § 2173.05(e), "explicit antecedent basis" is not required where, as in the instant application, the claims are not unclear, the Examiner respectfully submits that the claims are not rejected under 35 U.S.C. § 112, second paragraph, for insufficient antecedent basis. Examiner acknowledges that the scope of the claims would be reasonably ascertainable by those skilled in the art. However, the claims are not

rejected due to its substance but rather objected to due to its form, which the Examiner believes may be improved upon.

Therefore, for at least the reasons set forth above, the objections to Claims 4, 5, 8, 11-19, 21, and 25 are proper and therefore, maintained.

In the Remarks, Applicant argues:

b) The Office Action cites Sundararajan, which discusses a system having "components" that are a combination of hardware and software. See paragraph [0041]. There are multiple parallel subsystems, each have components, where components between the subsystems may have a particular relationship. See, inter alia, paragraphs [0038], [0041], [0042]. The relationships between the "components" are monitored and apparently managed. See, inter alia, paragraphs [0042], [0044]. Nowhere in the reference does Sundararajan disclose or suggest the instantiation of a software object from one software subsystem to another software subsystem. In fact, rather than suggesting changes to software objects that are instantiated in other subsystems, the whole of the Sundararajan reference appears to suggest changes to "components," meaning a change in a combination of a software elements with a hardware element. Thus, the cited reference appears to disclose nothing more than changing what software is associated with particular hardware on a subsystem, which may affect another subsystem that relied on a relationship of one of its components to the changed component.

Examiner's response:

b) Examiner disagrees. Applicant's arguments are not persuasive for at least the following reasons:

First, with respect to the Applicant's assertion that nowhere in the reference does Sundararajan disclose or suggest the instantiation of a software object from one software subsystem to another software subsystem, as previously pointed out in the Non-Final Rejection (mailed on 06/06/2008) and further clarified hereinafter, the Examiner respectfully submits that Sundararajan clearly discloses "wherein the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem" (*see Paragraph [0029], "In an example of a component change management system according to the present invention, an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features (the software object is instantiated in a second software subsystem to interact with software objects of the second software subsystem)."; Paragraph [0046], "After installing and/or revising the component at step 160, the component change manager 12, at step 170, communicates with the e-commerce server 14a, 14b and/or 14c, which hosts the production environment that received the component revision, to verify that the component revision is compatible with the previously defined system component interrelationships."*). As acknowledged by the Applicant on page 9 of the "Remarks," Sundararajan discloses that there are multiple parallel subsystems, each subsystem has components, where components between the subsystems may have a particular relationship. Note that Sundararajan further discloses that an e-commerce application can include a variety of device and application modules, combined in multiple configurations over a range of version levels with a variety of enabled features. Thus,

one of ordinary skill in the art would readily comprehend that an application module with a feature enabled in one configuration may also have the same or a different feature enabled in another configuration (instantiated in a second software subsystem).

Second, with respect to the Applicant's assertion that the whole of the Sundararajan reference appears to suggest changes to "components," meaning a change in a combination of a software element with a hardware element, the Examiner respectfully submits that Sundararajan also describes "components" as related to software components in other embodiments. For example, Sundararajan describes "components" as related to software components in the following portions with emphasis added:

- "A change in a component, such as a software application, in the production environment is checked by the component change system for compatibility" (*see Paragraph [0012]*).
- "Representative components may include hardware, application modules and language interpreters" (*see Paragraph [0016]*).
- "Changes introduced to components existing in the production environment can include new software installations, software upgrades, hardware installations and hardware upgrades" (*see Paragraph [0020]*).
- "Thus, verifying and maintaining the integrity of the application configuration definition is necessary before and after upgrading the e-commerce application with changes to the associated web application server software" (*see Paragraph [0023]*).
- "Other exemplary opportunities for verification of configuration integrity can include 1) after an initial installation of a complete application, 2) installation of one or more

components requiring changes within the context of the application, 3) a partial or complete downgrade of one or more components (e.g., such as a hardware device or software application) to a previous version of a system component, or 4) a partial or complete upgrade of one or more components to a later component version” (see *Paragraph [0023]*).

- “Then, prior to performing an upgrade or a change operation, e.g., a hardware reconfiguration, software upgrade or user driven redefinition of content or data; the integrity of the overall production environment is validated by identifying dependency incompatibilities of the relationships” (see *Paragraph [0026]*).

- “Although alternatives exist for identifying incompatibilities, a preferred method reads resident component attribute data (e.g. software version number) and new component attribute data (e.g. software version number), verifies compatibility between resident and new components by comparing current attribute data against the dedicated control field” (see *Paragraph [0027]*).

- “The data fields are encoded control fields identifying the compatibility of one component (e.g., web server software) with another component (e.g., web application software)” (see *Paragraph [0044]*).

Therefore, for at least the reasons set forth above, the rejections made under 35 U.S.C. § 102(b) with respect to Claims 1 and 32 and 35 U.S.C. § 103(a) with respect to Claims 6 and 22 are proper and therefore, maintained.

Conclusion

16. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

17. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191